Lab Exercise 2

Please follow the instruction in Workshop Note 2 to complete this exercise.

1. Turn on your computer and startup **Windows XP** (*English version*), download and install **J2ME Wireless Toolkit 2.2** to your computer. Then download the laboratory resource files from http://www.peter-lo.com/Teaching/I123-1-A/Source2.zip.

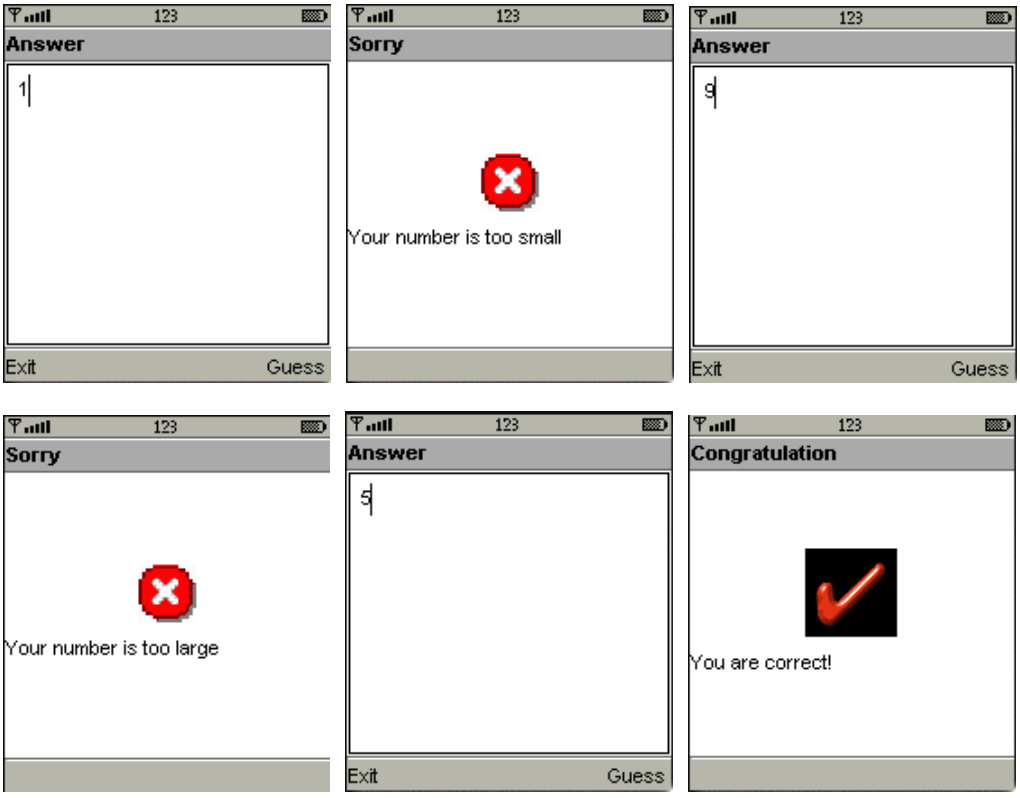
2. Create your first J2ME program with "Textbox", and also a **[Exit]** Command to allow user quit the program. Then create a command menu to allow user select. *(Page 1 – 6)*

3. Create a MIDlet with "List" object, then compare the different between "**EXCLUSIVE**" and "**MULTIPLE**" and "**IMPLICIT**". Which type of list allows user interaction? Then modify your MIDlet to display **Alert** with different alert sound. Finally, try to add the image to the alert and list. (*Page 5 – 18*)

4. Create a MIDlet to display horizontal scrolling banner by using **Ticker**. (*Page 19 – 20*)

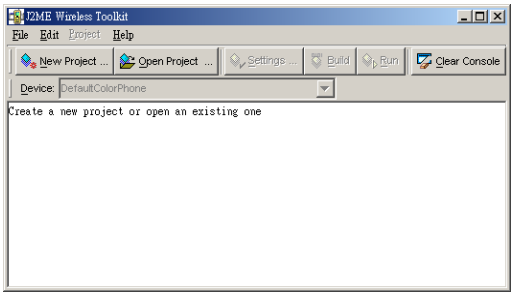
5. Create a MIDlet to pick a number randomly and display it on screen. (*Page 21 – 22*)

6. Now it's time for you to create the first game: "***Guess Number***". The program should pick a secret number (0 – 9) and let the user guess what number it is. If the guess is too high or too low, the program should provide a hint. You can use Ticker component to show the instruction, and use the Alert component to display the message.

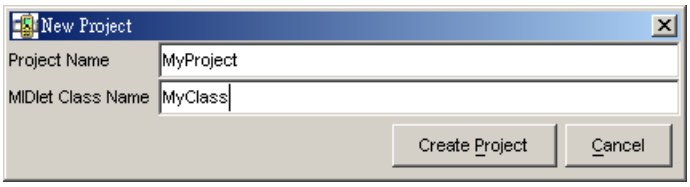


1. Create a Textbox

1. Start "***KToolbar***" in "***J2ME Wireless Toolkit 2.2***" program group.



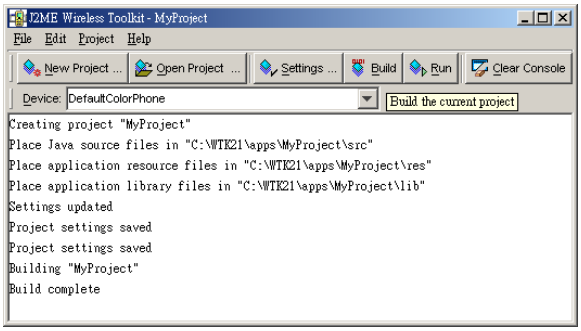
2. Use **[New Project…]** to create a new project. Let's name the project at "***MyProject***" and the class at "***MyClass***". Then click **[Create Project]** to continue.



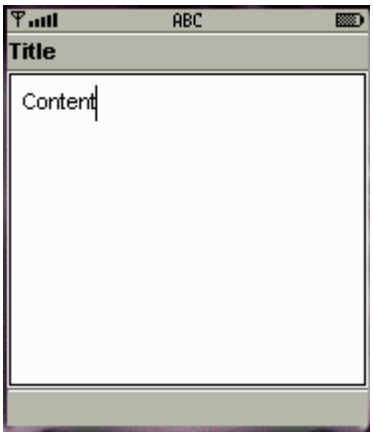
3. Use "Notepad" or other Text Editor to create a file called "***MyClass.java***" in the "***src***" folder under your project folder (C:\WTK22\apps\MyProject\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet {
  // Initialize the Midlet Display variable
  private Display midletDisplay;

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Create the TextBox containing the "Content" message
    TextBox mainScreen = new TextBox("Title", "Content", 50, TextField.ANY);

    // Set the current display of the midlet to the textBox screen
    midletDisplay.setCurrent(mainScreen);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }
}
```

4. Use the **[Build]** button to build the project.



5. If there is no error occurs, you can use **[Run]** button to test the program.

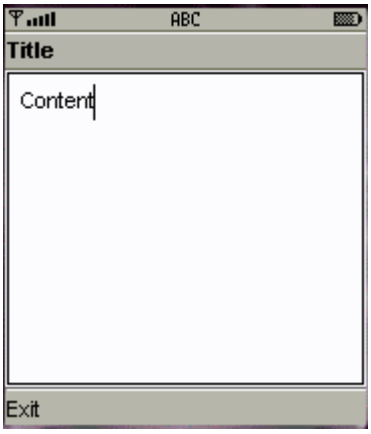


2. Create a Command Button

1. Start "***Ktoolbar***" in "***J2ME Wireless Toolkit 2.2***". Then open the project "***MyProject***" you created before.
2. Modify the source code for your java file "***MyClass.java***" as follow:

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Initialize the Midlet Display variable
  private Display midletDisplay;
  private Command cmdExit;

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);

    // Define the Exit Button with "Exit" label
    cmdExit = new Command("Exit", Command.EXIT, 0);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Create the TextBox containing the "Content" message
    TextBox mainScreen = new TextBox("Title", "Content", 50, TextField.ANY);

    // Create the Exit Button
    mainScreen.addCommand(cmdExit);

    // Add the Command Listener
    mainScreen.setCommandListener(this);

    // Set the current display of the midlet to the textBox screen
    midletDisplay.setCurrent(mainScreen);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

3. Use the **[Build]** button to build the project, then press **[Run]** button to test the program. A new **[Exit]** button is display at the left bottom.



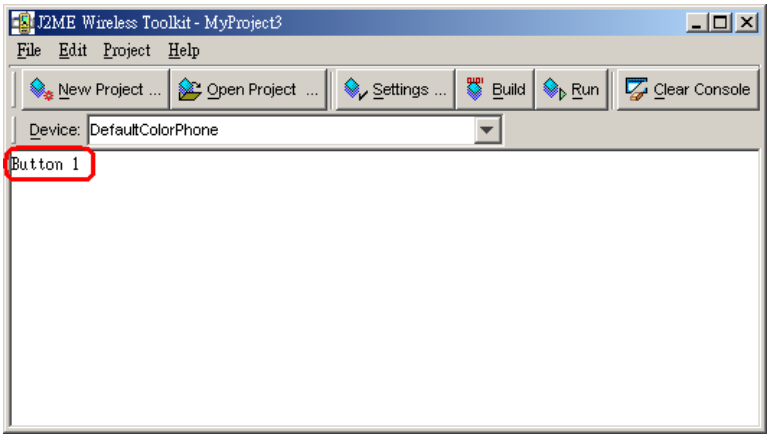
3. Create a Multiple Command Menu

1. Open the project "**MyProject**" you created before and modify the source code for your java file "**MyClass.java**" as follow:

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Initialize the Midlet Display variable
  private Display midletDisplay;
  private Command cmdExit;
  private Command mCommand1, mCommand2, mCommand3, mCommand4, mCommand5;

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);

    // Define the Exit Button with "Exit" label
    cmdExit = new Command("Exit", Command.EXIT, 0);

    // Create the Multiple Button Menu with different prior
    mCommand1 = new Command("Button1", Command.SCREEN, 0);
    mCommand2 = new Command("Button2", Command.SCREEN, 1);
    mCommand3 = new Command("Button3", Command.SCREEN, 2);
    mCommand4 = new Command("Button4", Command.SCREEN, 3);
    mCommand5 = new Command("Button5", Command.SCREEN, 4);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Create the TextBox containing the "Content" message
    TextBox mainScreen = new TextBox("Title", "Content", 50, TextField.ANY);

    // Create the Exit Button
    mainScreen.addCommand(cmdExit);
    mainScreen.addCommand(mCommand1);
    mainScreen.addCommand(mCommand2);
    mainScreen.addCommand(mCommand3);
    mainScreen.addCommand(mCommand4);
    mainScreen.addCommand(mCommand5);

    // Add the Command Listener
    mainScreen.setCommandListener(this);

    // Set the current display of the midlet to the textBox screen
    midletDisplay.setCurrent(mainScreen);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }
```

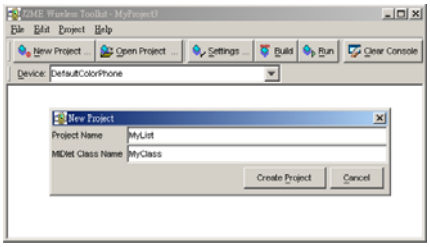
```
  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    } else if (c == mCommand1) {
      System.out.println("Button 1");
    } else if (c == mCommand2) {
      System.out.println("Button 2");
    } else if (c == mCommand3) {
      System.out.println("Button 3");
    } else if (c == mCommand4) {
      System.out.println("Button 4");
    } else if (c == mCommand5) {
      System.out.println("Button 5");
    }
  }
}
```

2. Use the **[Build]** button to build the project, then press **[Run]** button to test the program. You can find a multiple menu displayed on your screen. The message “Button 1” will display on the console screen after you press **[Select]** to confirm “Button 1”



4. Create a Implicit List

1. Start "***KToolbar***" in "***J2ME Wireless Toolkit 2.2***" program group. Then use **[New Project…]** to create a new project. Let's name the project at "***MyImplicitList***" and the class at "***MyClass***". Then click ***[Create Project]*** to continue.

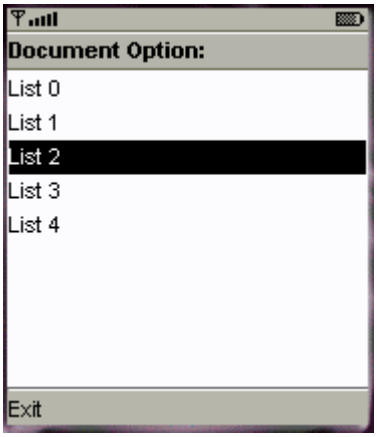


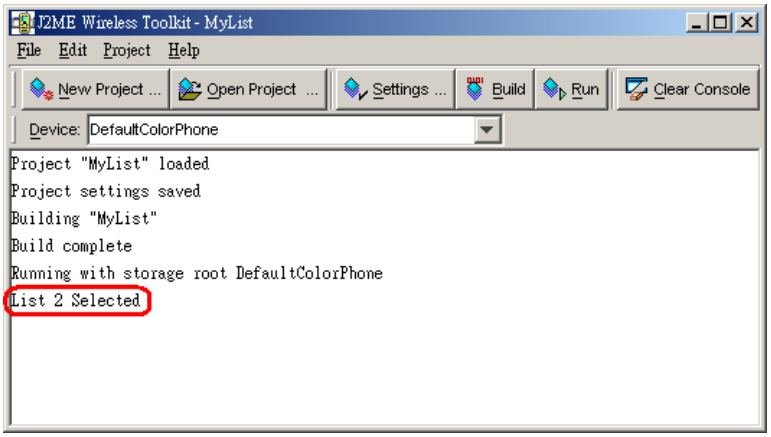
2. Create a java source file called "***MyClass.java***" in the "***src***" folder under your project folder (C:\WTK22\apps\MyImplicitList\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Initialize the Midlet Display variable
  private Display midletDisplay;    // Main Display
  private Command cmdExit;      // Exit Button
  private List ImplicitList;        // Main List

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);
    cmdExit = new Command("Exit", Command.EXIT, 0);

    // Create array of corresponding string objects
    String StringList[] = {"List 0", "List 1", "List 2", "List 3", "List 4"};

    // Create list using arrays, add commands, listen for events
    ImplicitList = new List("Document Option:", List.IMPLICIT, StringList, null);
    ImplicitList.addCommand(cmdExit);
    ImplicitList.setCommandListener(this);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Set the current display of the midlet to the list screen
    midletDisplay.setCurrent(ImplicitList);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }
```

```
  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    // If an implicit list generated the event
    if (c == List.SELECT_COMMAND) {
      switch (ImplicitList.getSelectedIndex()) {
        case 0:
          System.out.println("List 0 Selected");
          break;
        case 1:
          System.out.println("List 1 Selected");
          break;
        case 2:
          System.out.println("List 2 Selected");
          break;
        case 3:
          System.out.println("List 3 Selected");
          break;
        case 4:
          System.out.println("List 4 Selected");
          break;
      }
    } else if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

3. Use the **[Build]** button to build the project, then press **[Run]** button to test the program. You can find a list displayed on your screen. Select the item to obtain the message on the console.



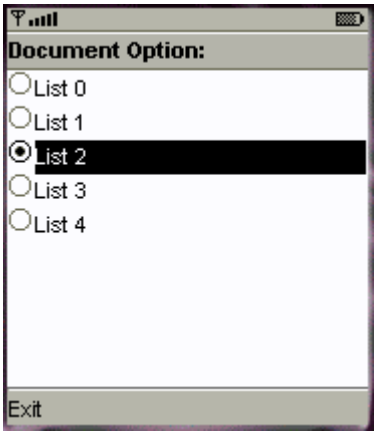


5. Create a Exclusive List (Radio Button)

1. Start "***KToolbar***" in "***J2ME Wireless Toolkit 2.2***" program group. Then use **[New Project…]** to create a new project. Let's name the project at "***MyExclusiveList***" and the class at "***MyClass***". Then click **[Create Project]** to continue.
2. Create a java source file called "***MyClass.java***" in the "***src***" folder under your project folder (C:\WTK22\apps\MyExclusiveList\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Initialize the Midlet Display variable
  private Display midletDisplay;    // Main Display
  private Command cmdExit;          // Exit Button
  private List ExclusiveList;       // Main List

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);
    cmdExit = new Command("Exit", Command.EXIT, 0);

    // Create array of corresponding string objects
    String StringList[] = {"List 0", "List 1", "List 2", "List 3", "List 4"};

    // Create list using arrays, add commands, listen for events
    ExclusiveList = new List("Document Option:", List.EXCLUSIVE, StringList, null);
    ExclusiveList.addCommand(cmdExit);
    ExclusiveList.setCommandListener(this);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Set the current display of the midlet to the list screen
    midletDisplay.setCurrent(ExclusiveList);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

3. Use the **[Build]** button to build the project, then press **[Run]** button to test the program. The outlook for the list will be changed.

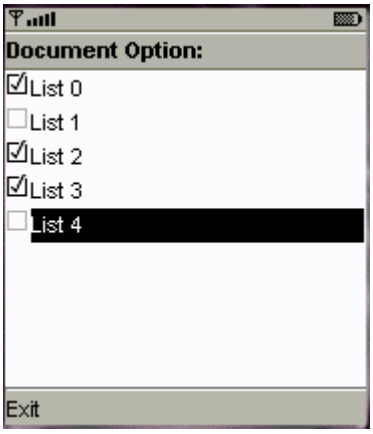


6. Create a Multiple List (Check Box)

1. Start "***KToolbar***" in "***J2ME Wireless Toolkit 2.2***" program group. Then use **[*New Project…*]** to create a new project. Let's name the project at "***MyMultipleList***" and the class at "***MyClass***". Then click **[*Create Project*]** to continue.
2. Create a java source file called "***MyClass.java***" in the "***src***" folder under your project folder (C:\WTK22\apps\MyMultipleList\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Initialize the Midlet Display variable
  private Display midletDisplay;    // Main Display
  private Command cmdExit;          // Exit Button
  private List MultipleList;        // Main List

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);
    cmdExit = new Command("Exit", Command.EXIT, 0);

    // Create array of corresponding string objects
    String StringList[] = {"List 0", "List 1", "List 2", "List 3", "List 4"};

    // Create list using arrays, add commands, listen for events
    MultipleList = new List("Document Option:", List.MULTIPLE, StringList, null);
    MultipleList.addCommand(cmdExit);
    MultipleList.setCommandListener(this);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Set the current display of the midlet to the list screen
    midletDisplay.setCurrent(MultipleList);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

3. Use the **[Build]** button to build the project, then press **[Run]** button to test the program. You can use **[Select]** button and the cursor to select multiple lists.



7. Add an Immutable Image in a List

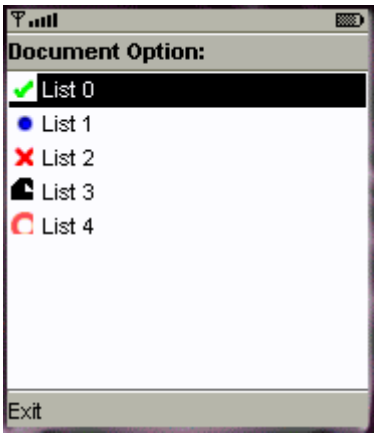
1. Start "***KToolbar***" in "***J2ME Wireless Toolkit 2.2***" program group. Then use **[*New Project…*]** to create a new project. Let's name the project at "***MyImageList***" and the class at "***MyClass***". Then click **[*Create Project*]** to continue.
2. Copy the following PNG file to the resource folder (C:\WTK22\apps\MyImageList\res).

3. Create a java source file called "***MyClass.java***" in the "***src***" folder under your project folder (C:\WTK22\apps\MyImageList\src). You can also copy and modify the source file from the project "MyImplicitList".

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Initialize the Midlet Display variable
  private Display midletDisplay;    // Main Display
  private List ImplicitList;        // Main List
  private Command mExitCommand;     // Exit Button

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);
    mExitCommand = new Command("Exit", Command.EXIT, 0);

    // Create array of corresponding string objects
    String StringList[] = {" List 0", " List 1", " List 2", " List 3", " List 4"};

    // Create array of image objects
    try {
      Image ImageList[] = { Image.createImage("/list0.png"),
                            Image.createImage("/list1.png"),
                            Image.createImage("/list2.png"),
                            Image.createImage("/list3.png"),
                            Image.createImage("/list4.png") };

      // Create list using arrays, add commands, listen for events
      ImplicitList = new List("Document Option:", List.IMPLICIT, StringList, ImageList);
      ImplicitList.addCommand(mExitCommand);
      ImplicitList.setCommandListener(this);
    } catch (java.io.IOException e) {
      System.err.println("Unable to locate or read .png file");
    }
  }

  // Create the Hello Midlet World TextBox
  public void startApp() {
    // Set the current display of the midlet to the textBox screen
    midletDisplay.setCurrent(ImplicitList);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
```

```
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    // If an implicit list generated the event
    if (c == List.SELECT_COMMAND) {
      switch (ImplicitList.getSelectedIndex()) {
        case 0:
          System.out.println("List 0 Selected");
          break;
        case 1:
          System.out.println("List 1 Selected");
          break;
        case 2:
          System.out.println("List 2 Selected");
          break;
        case 3:
          System.out.println("List 3 Selected");
          break;
        case 4:
          System.out.println("List 4 Selected");
          break;
      }
    } else if (c == mExitCommand) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

4. Compile and execute the MIDlet. You can see your graphical implicit list as follow.



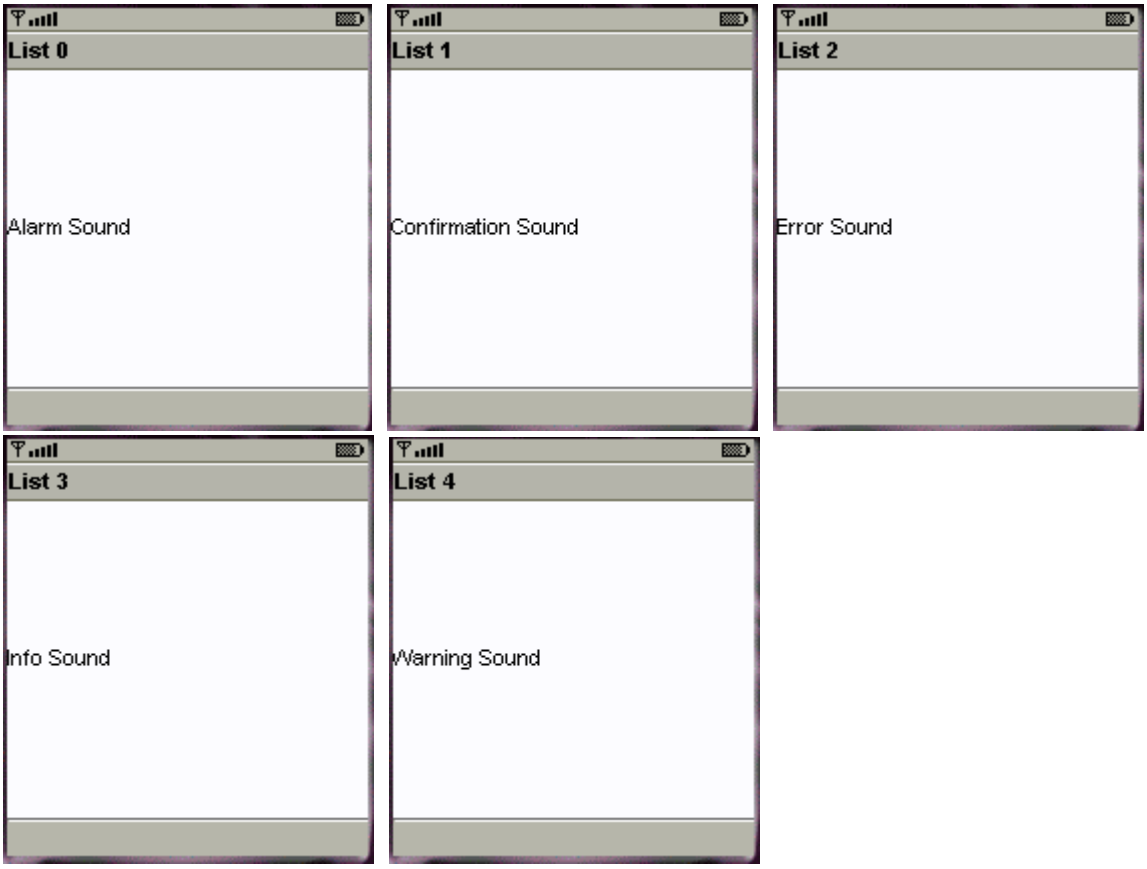
8. Display an Alert Message

1. Open the project "***MyImplicitList***" you created before and modify the source code for your java file "***MyClass.java***" as follow:

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Initialize the Midlet Display variable
  private Display midletDisplay;    // Main Display
  private Command cmdExit;          // Exit Button
  private List ImplicitList;        // Main List

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);
    cmdExit = new Command("Exit", Command.EXIT, 0);

    // Create array of corresponding string objects
    String StringList[] = {"List 0", "List 1", "List 2", "List 3", "List 4"};

    // Create list using arrays, add commands, listen for events
    ImplicitList = new List("Document Option:", List.IMPLICIT, StringList, null);
    ImplicitList.addCommand(cmdExit);
    ImplicitList.setCommandListener(this);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Set the current display of the midlet to the list screen
    midletDisplay.setCurrent(ImplicitList);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    // If an implicit list generated the event
    if (c == List.SELECT_COMMAND) {
      // Construct the Alert
      Alert al = null;

      switch (ImplicitList.getSelectedIndex()) {
        case 0:
          al = new Alert("List 0", "Alarm Sound", null, AlertType.ALARM);
          break;
        case 1:
          al = new Alert("List 1", "Confirmation Sound", null, AlertType.CONFIRMATION);
          break;
```

```
        case 2:
          al = new Alert("List 2", "Error Sound", null, AlertType.ERROR);
          break;
        case 3:
          al = new Alert("List 3", "Info Sound", null, AlertType.INFO);
          break;
        case 4:
          al = new Alert("List 4", "Warning Sound", null, AlertType.WARNING);
          break;
      }
      // Display the Alert on screen
      midletDisplay.setCurrent(al);
    } else if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

2. Use the **[Build]** button to build the project, then press **[Run]** button to test the program. You can listen the alert sound if you have turn on your speaker.



9. Add an Immutable Image in a Alert

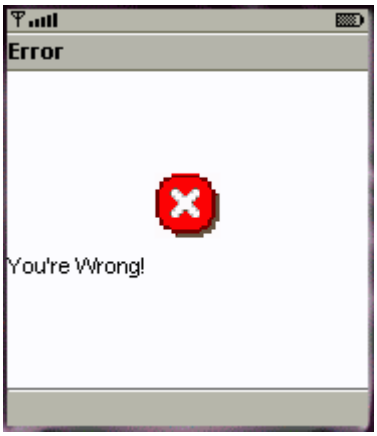
1. Create a new project, name the project and the class at "***MyImageAlert***" and "***MyClass***". Then copy the follow picture to the resource folder (C:\WTK22\apps\MyImageAert\src).

2. Create a java source file called "***MyClass.java***" in the "***src***" folder under your project folder (C:\WTK22\apps\MyImageAlert\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                 */
/*************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  private Display midletDisplay;    // Reference to Display object
  private Command cmdView;          // Command Button to view the Alert
  private Command cmdExit;          // Command Button to exit
  private TextBox mainScreen;       // Text Box to store Answer


  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);

    // Create the View and Exit button
    cmdExit = new Command("Exit", Command.EXIT, 0);
    cmdView = new Command("View", Command.SCREEN, 1);

    // Create Textbox to store answer
    mainScreen = new TextBox ("Answer", "", 8, TextField.NUMERIC);

    // Create the View and Exit Button
    mainScreen.addCommand(cmdExit);
    mainScreen.addCommand(cmdView);

    // Add the Command Listener
    mainScreen.setCommandListener(this);

    // Set the current display of the midlet to the textBox screen
    midletDisplay.setCurrent(mainScreen);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Set the current display of the midlet to the Form
    midletDisplay.setCurrent(mainScreen);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
```

```
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdView) {
      try {
        // Read the appropriate image
        Image im = Image.createImage("/error.png");

        // Output Alert
        Alert al = new Alert("Error", "You're Wrong!", im, AlertType.CONFIRMATION);

        // Display the Alert on screen
        midletDisplay.setCurrent(al);
      } catch (java.io.IOException e) {
          System.err.println("Unable to locate or read the PNG file");
      }
    } else if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

3. Compile and execute the MIDlet. Press **[View]** button to display the alert.

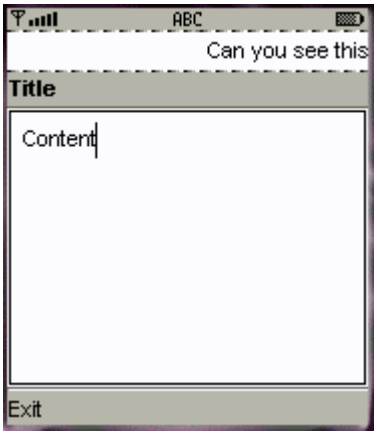


10. Display an Ticker

1. Within your "***J2ME Wireless Toolkit 2.2***", use **[New Project…]** to create a new project. Let's name the project at "***MyTicker***" and the class at "***MyClass***".
2. Create a java source file called "***MyClass.java***" in the "***src***" folder under your project folder (C:\WTK22\apps\MyTicker\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Initialize the Midlet Display variable
  private Display midletDisplay;    // Display
  private Command cmdExit;      // Command
  private Ticker myTicker;          // Ticker

  // Define the no-argument constructor
  public MyClass() {
    midletDisplay = Display.getDisplay(this);
    cmdExit = new Command("Exit", Command.EXIT, 0);
    myTicker = new Ticker("Can you see this Ticker on your screen?");
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Create the TextBox containing the "Hello Midlet World!!" message
    TextBox mainScreen = new TextBox("Title", "Content", 50, TextField.ANY);

    mainScreen.setTicker(myTicker);          // Set the ticker
    mainScreen.addCommand(cmdExit);      // Add the Exit command
    mainScreen.setCommandListener(this);     // Add the Command Listener

    // Set the current display of the midlet to the textBox screen
    midletDisplay.setCurrent(mainScreen);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

3. Use the **[Build]** button to build the project, then press **[Run]** button to test the program. Can you see the horizontal scrolling banner?



11. Generate a Random Number

1. In your "***J2ME Wireless Toolkit 2.2***", use **[New Project…]** to create a new project. Let's name the project at "***MyRandom***" and the class at "***MyClass***".
2. Create a java source file called "***MyClass.java***" in the "***src***" folder under your project folder (C:\WTK22\apps\MyRandom\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include Java Random utility
import java.util.Random;

/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  private Display midletDisplay;        // Reference to Display object
  private Command cmdExit;              // Command to exit the MIDlet
  private Random rand = new Random();   // Initialize Random Seek
  private int RandomNumber = 0;         // Random Number
  private String str = null;            // Output String

  // Define the no-argument constructor
  public MyClass() {
    // Retrieve the display from the static display object
    midletDisplay = Display.getDisplay(this);

    // Define the Exit Button with "Exit" label
    cmdExit = new Command("Exit", Command.EXIT, 0);
  }

  // Called by application manager to start the MIDlet.
  public void startApp() {
    // Random a number between 0 to 9
    RandomNumber = Math.abs(rand.nextInt())% 9;

    // Store the number to the output string
    str = "The number is " + RandomNumber;

    // Create the TextBox containing the random number
    TextBox mainScreen = new TextBox("Random Number", str, 50, TextField.ANY);

    // Create the Exit Button
    mainScreen.addCommand(cmdExit);

    // Add the Command Listener
    mainScreen.setCommandListener(this);

    // Set the current display of the midlet to the textBox screen
    midletDisplay.setCurrent(mainScreen);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }
```

```
  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
     if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

3. Use the **[Build]** button to build the project, then press **[Run]** button to test the program. You can generate a number randomly, and then press **[Exit]** to quit and execute the program again, can you generate another number?